Introducción a las MLOps

Presentación con una selección de las slides de las dos primeras presentaciones de Chip Huyen que se pueden consultar en stanford-cs329s.github.io/syllabus.html

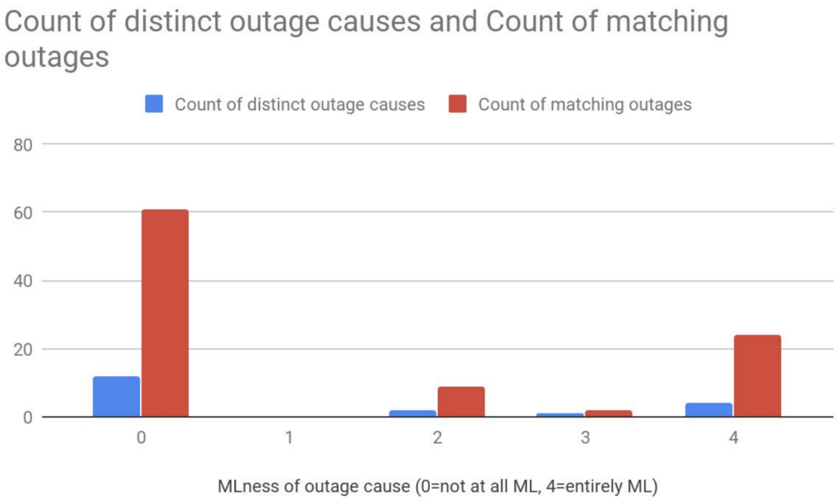
Why ML Systems Design?

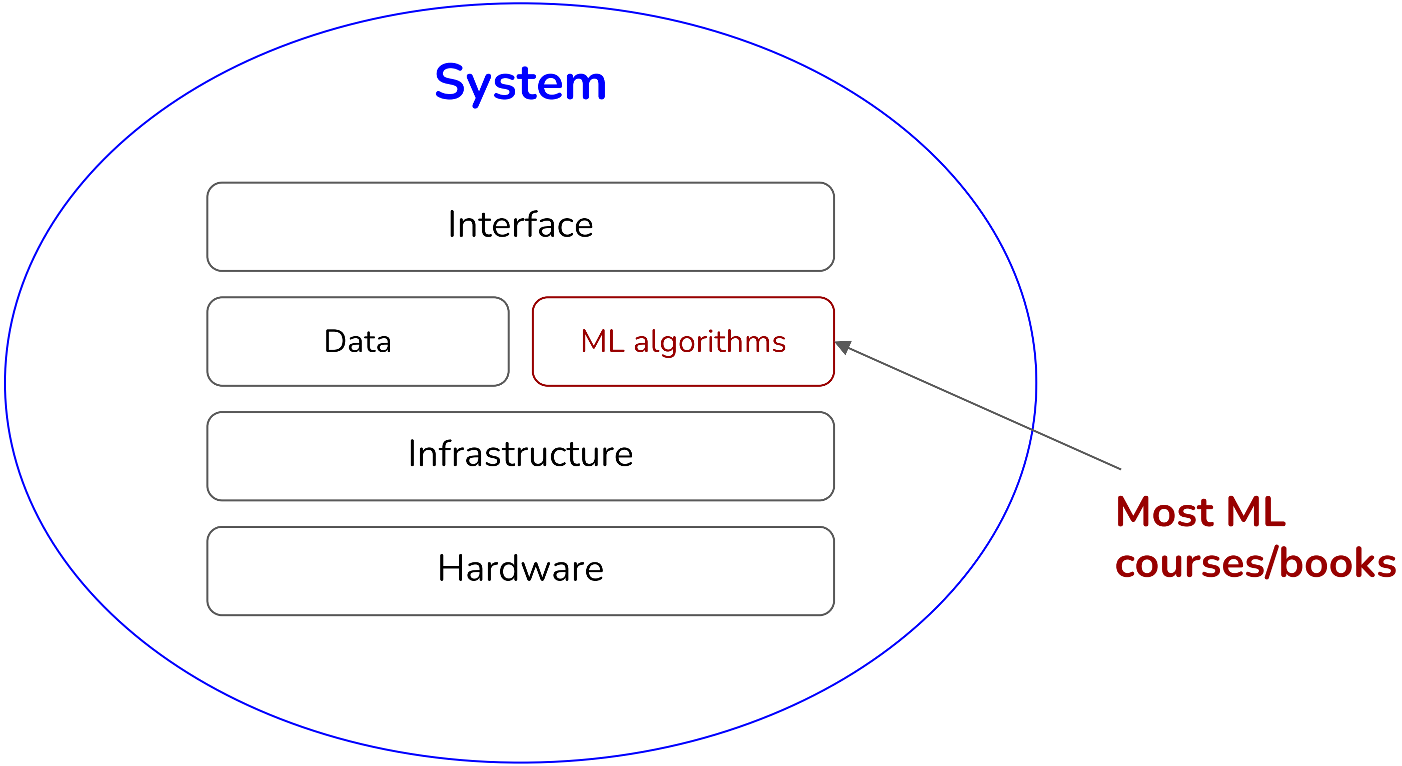
- ML algorithms is the less problematic part.
- The hard part is to **how to make algorithms work with other parts to solve real-world problems**.

Why ML Systems Design?

- ML algorithms is the less problematic part.
- The hard part is to **how to make algorithms work with other parts to solve real-world problems**.
- 60/96 failures caused by non-ML components

More on ML systems failures later!




System
Interface
Data
ML algorithms
Infrastructure
Hardware
Most ML
courses/books

What’s machine learning systems design?

The process of defining the **interface, algorithms, data**, **infrastructure**, and **hardware** for a machine learning system to satisfy **specified requirements**.

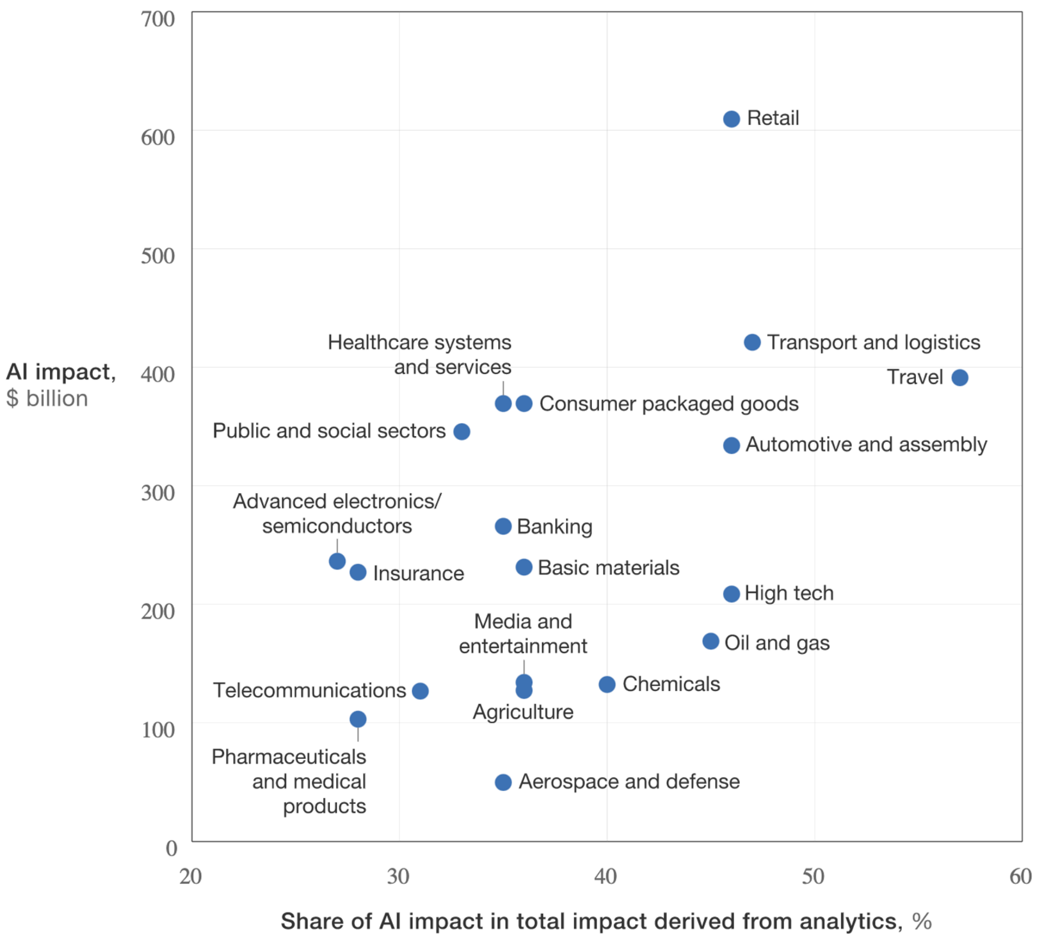
What's machine learning systems design?

The process of defining the **interface, algorithms, data**, **infrastructure**, and **hardware** for a machine learning system to satisfy **specified requirements**.

$\overbrace{\hphantom{xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx}}^{}$

reliable, scalable, maintainable, adaptable

Artificial intelligence (AI) has the potential to create value across sectors.



McKinsey&Company | **Source:** McKinsey Global Institute analysis

AI value creation by 2030

13 trillion USD

Most of it will be outside the consumer internet industry

We need more people from non-CS background in AI!

ML research vs. ML production

	Research	Production
Objectives	Model performance*	Different stakeholders have different objectives

"*" It's actively being worked. See Utility is in the Eye of the User: A Critique of NLP Leaderboards (Ethayarajh and Jurafsky, EMNLP 2020)

Leaderboard-style ML

- More comprehensive utility function
  - Model performance (e.g. accuracy)
  - Latency
  - Prediction cost
  - Interpretability
  - Robustness
  - Ease of use (e.g. OSS tools, community support)
  - Hardware requirements
- Adaptive to different use cases
  - Instead of a leaderboard for each dataset/task, the leaderboard adapts to each company's needs
- Dynamic datasets
  - Realistic distribution shifts with different types of shifts

Computational priority

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency

generating predictions

Latency matters

- 100ms delay can hurt conversion rates by 7% (Akamai study ‘17)
- 30% increase in latency costs 0.5% conversion rate (Booking.com ‘19)
- 53% phone users will leave a page that takes >3s to load (Google ‘16)

ML in research vs. in production

	Research	**Production**
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting

Data

Research	**Production**
● Clean ● Static ● Mostly historical data	● Messy ● Constantly shifting ● Historical + streaming data ● Biased, and you don’t know how biased ● Privacy + regulatory concerns

THE COGNITIVE CODER

By Armand Ruiz, Contributor, InfoWorld | SEP 26, 2017 7:22 AM PDT

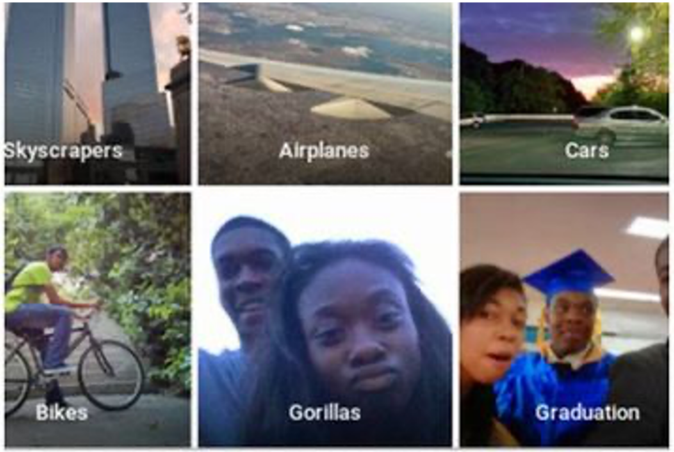
The 80/20 data science dilemma

Most data scientists spend only 20 percent of their time on actual data analysis and 80 percent of their time finding, cleaning, and reorganizing huge amounts of data, which is an inefficient data strategy

ML in research vs. in production

	Research	Production
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important

Fairness



Google Shows Men Ads for Better Jobs

by Krista Bradford | Last updated Dec 1, 2019

The Berkeley study found that both face-to-face and online lenders rejected a total of 1.3 million creditworthy black and Latino applicants between 2008 and 2015. Researchers said they believe the applicants "would have been accepted had the applicant not been in these minority groups." That's because when they used the income and credit scores of the rejected applications but deleted the race identifiers, the mortgage application was accepted.

ML in research vs. in production

	Research	Production
Objectives	Model performance	Different stakeholders have different objectives
Computational priority	Fast training, high throughput	Fast inference, low latency
Data	Static	Constantly shifting
Fairness	Good to have (sadly)	Important
Interpretability*	Good to have	Important

4. ML systems vs. traditional software

Traditional software

Separation of Concerns is a design principle for separating a computer program into distinct sections such that each section addresses a separate concern

- Code and data are separate
  - Inputs into the system shouldn't change the underlying code

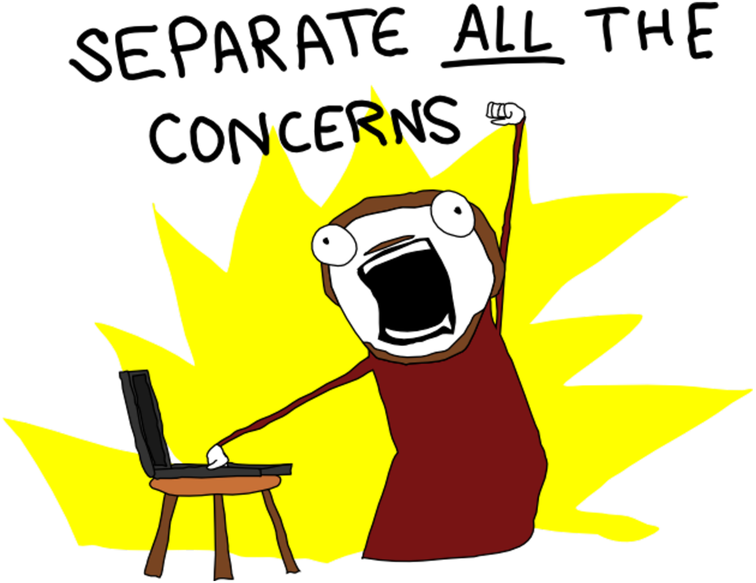


Image by [Arda Cetinkaya]

ML systems

- Code and data are tightly coupled
    - ML systems are part code, part data
- Not only test and version code, need to $\underbrace{\text{test and version data}}_{\text{the hard part}}$ too

ML systems: version data

- Line-by-line diffs like Git doesn't work with datasets
- Can't naively create multiple copies of large datasets
- How to merge changes?

How to ...

- Validate data correctness?
- Test features’ usefulness?
- Detect when the underlying data distribution has changed?
- Know if the changes are bad for models without ground truth labels?
- Detect malicious data?
    - Not all data points are equal (e.g. scans of cancerous lungs are more valuable)
    - Bad data might harm your model and/or make it susceptible to attacks

ML systems: data poisoning attacks

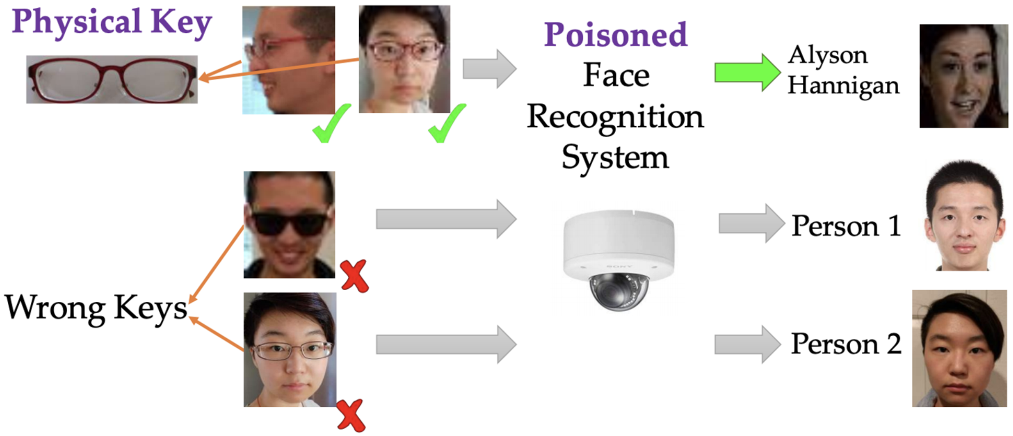


Fig. 1: An illustrating example of backdoor attacks. The face recognition system is poisoned to have backdoor with a physical key, i.e., a pair of commodity reading glasses. Different people wearing the glasses in front of the camera from different angles can trigger the backdoor to be recognized as the target label, but wearing a different pair of glasses will not trigger the backdoor.

Targeted Backdoor Attacks on Deep Learning Systems Using Data Poisoning (Chen et al., 2017)

Engineering challenges with large ML models

- Too big to fit on-device
- Consume too much energy to work on-device
- Too slow to be useful
    - Autocompletion is useless if it takes longer to make a prediction than to type
- If unit/CI tests take hours, the development cycles will stagnate

5. ML production myths

Myth #1: Deploying is hard

Myth #1: Deploying is hard

Deploying is easy. Deploying reliably is hard

Myth #2: You only deploy one or two ML models at a time

Myth #2: You only deploy one or two ML models at a time

Booking.com: 150+ models, Uber: thousands

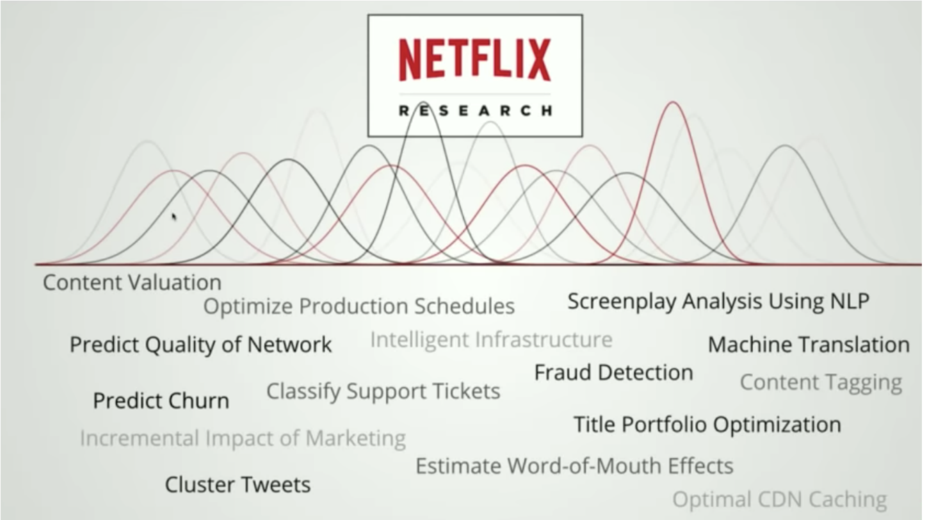


Image from Ville Tuulos (Netflix, Outerbounds)

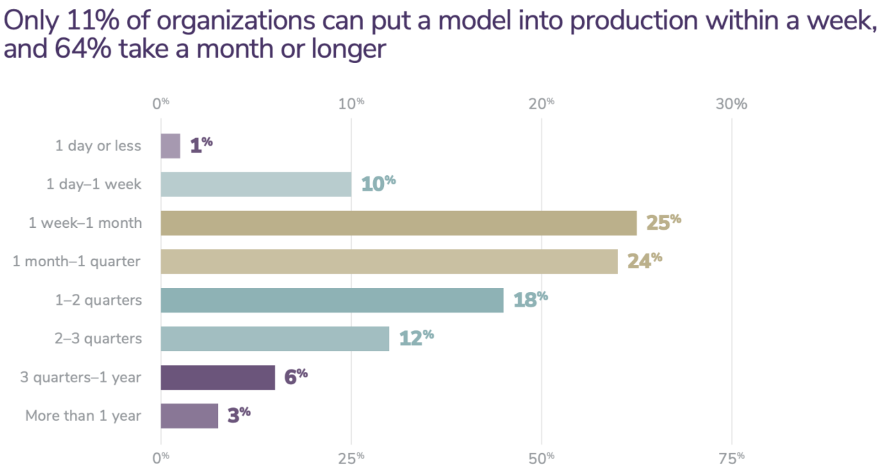
Myth #3: You won’t need to update your models as much

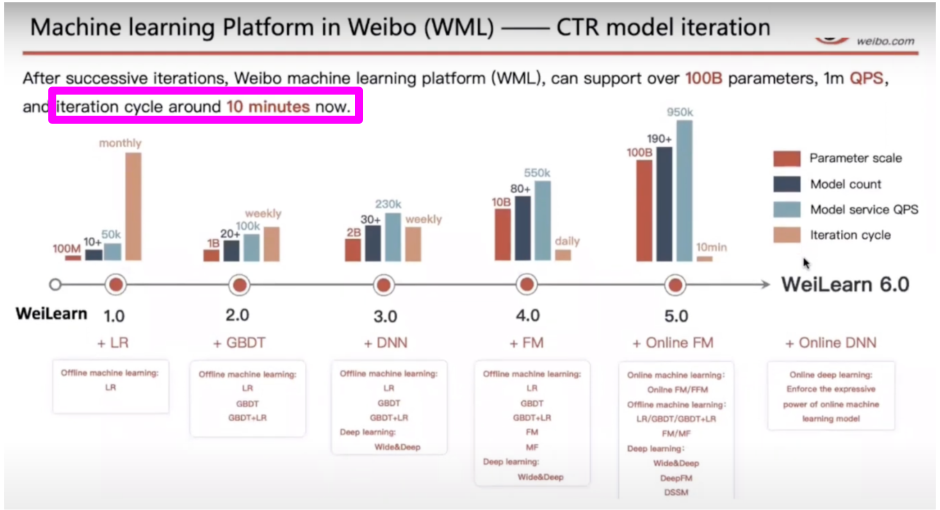
DevOps: Pace of software delivery is accelerating

- Elite performers deploy **973x** more frequently with **6570x** faster lead time to deploy (Google DevOps Report, 2021)
- DevOps standard (2015)
  - Etsy deployed 50 times/day
  - Netflix 1000s times/day
  - AWS every 11.7 seconds

DevOps to MLOps: Slow vs. Fast

We'll learn how to do minute-iteration cycle!





Left image from Algorithmia | Right image: [Machine learning with Flink in Weibo](#) (Qian Yu, QCon 2019)

Accelerating ML Delivery

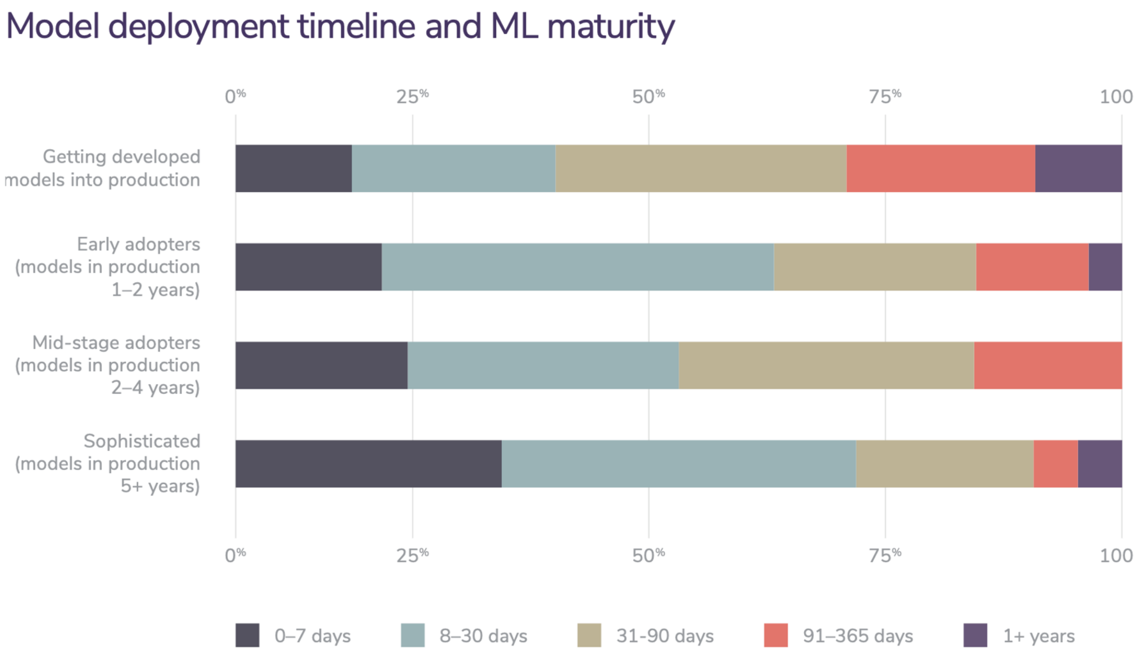
ML + DevOps = 🚀

Myth #4: ML can magically transform your business overnight

Myth #4: ML can magically transform your business overnight

Magically: possible
Overnight: no

Efficiency improves with maturity



[2020 state of enterprise machine learning]() (Algorithmia, 2020)

ML engineering is more engineering than ML

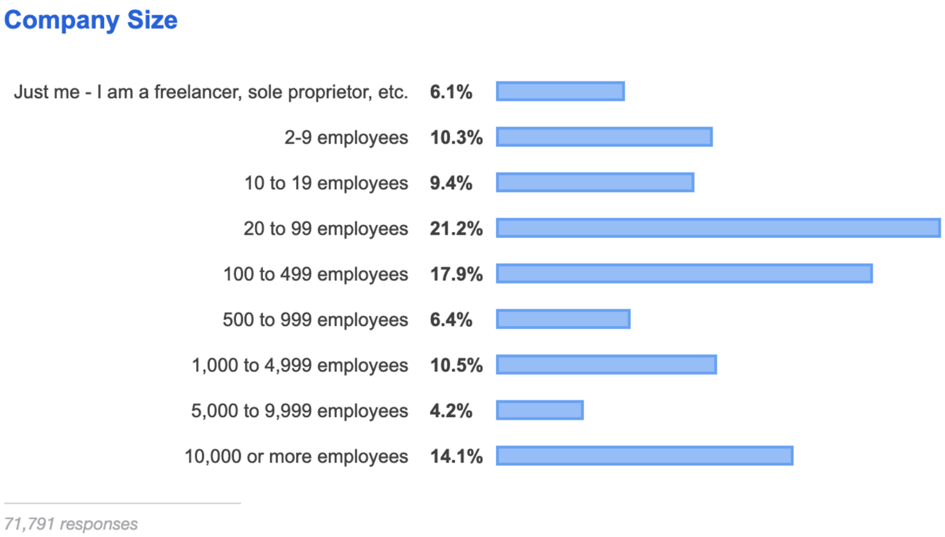
MLEs might spend most of their time:

- wrangling data
- understanding data
- setting up infrastructure
- deploying models

instead of training ML models

Myth #5: Most ML engineers don’t need to worry about scale

Myth #5: Most ML engineers don't need to worry about scale



[StackOverflow Developer Survey 2019]

1. ML Systems Fundamentals

ML in production: expectation

1. Collect data
2. Train model
3. Deploy model

ML in production: reality

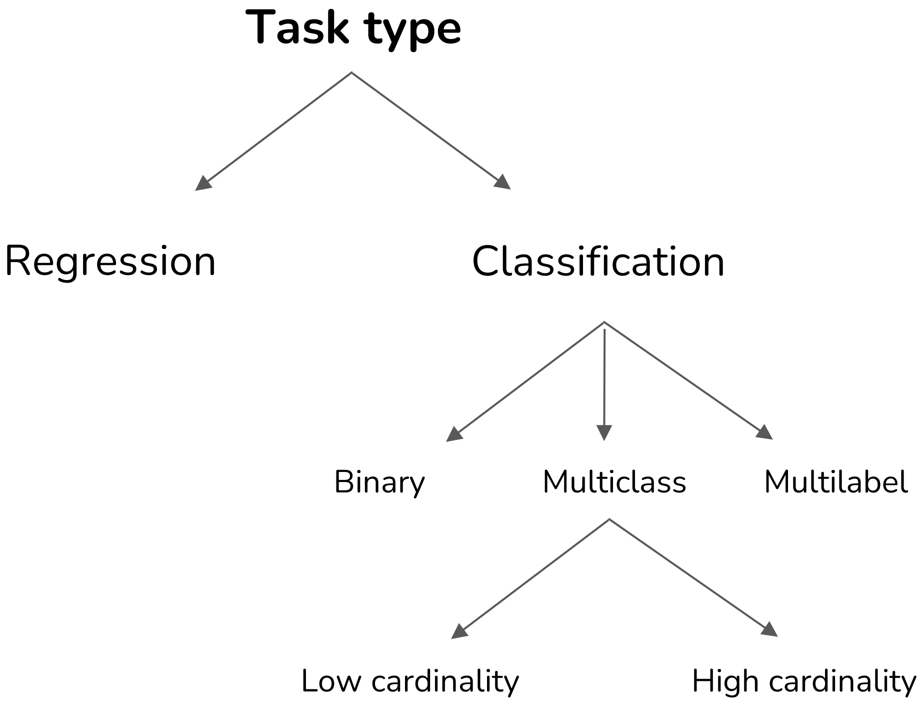
1. Choose a metric to optimize
2. Collect data
3. Train model
4. Realize many labels are wrong -> relabel data
5. Train model
6. Model performs poorly on one class -> collect more data for that class
7. Train model
8. Model performs poorly on most recent data -> collect more recent data
9. Train model
10. Deploy model
11. Dream about $$$
12. Wake up at 2am to complaints that model biases against one group -> revert to older version
13. Get more data, train more, do more testing
14. Deploy model
15. Pray
16. Model performs well but revenue decreasing
17. Cry
18. Choose a different metric
19. Start over

Step 15 and 17 are essential

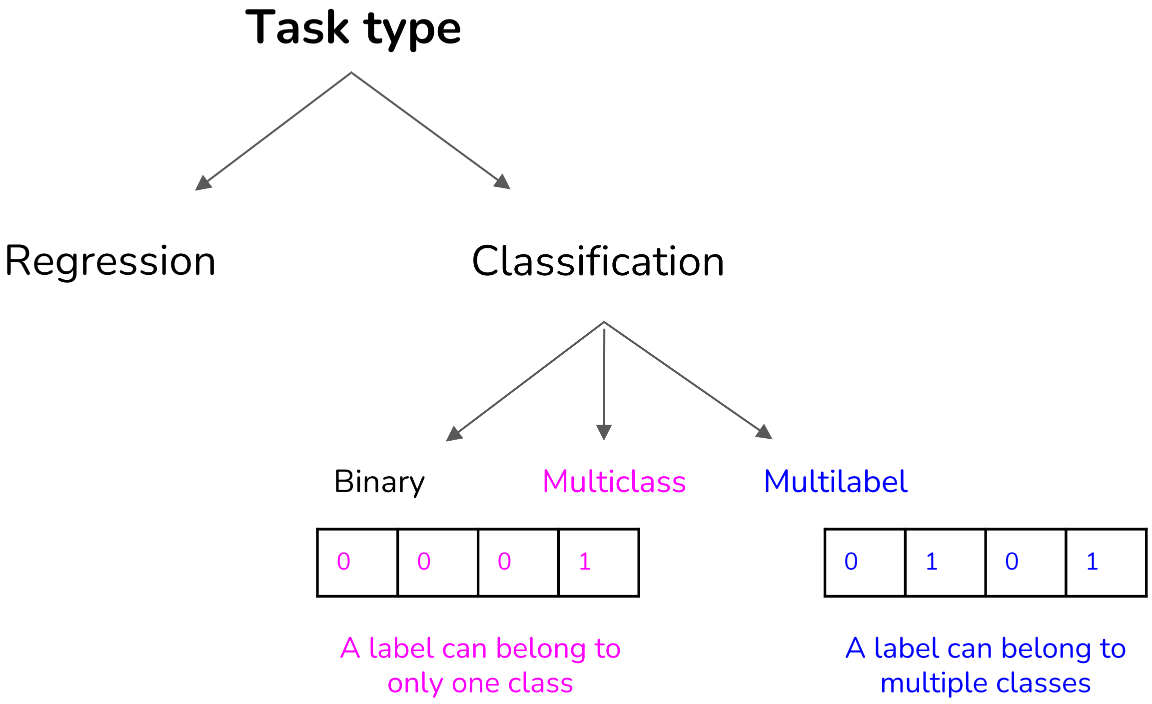
Project considerations

1. Framing
2. Objectives
3. Constraints
4. Phases

Framing the problem



Multiclass vs. multilabel



How to handle multilabel tasks

Multilabel problem solution

A multiclass problem

0	1	0	1

A set of multiple binary problems

Model 1: Does this belong to class 1?

Model 2: Does this belong to class 2?

...

Multilabel is harder than multiclass

Multilabel problem solution

A multiclass problem

0	1	0	1

A set of multiple binary problems

Model 1: Does this belong to class 1?

Model 2: Does this belong to class 2?

...

1. How to create ground truth labels?
2. How to decide decision boundaries

Multilabel: decision boundaries

Multilabel problem solution

A multiclass problem

0	1	2	3
0.45	0.33	0.2	0.02

Poll:
Which classes should this example belong to?

1. 0
2. 0, 1
3. 0, 1, 2

A set of multiple binary problems

Model 1: Does this belong to class 1?

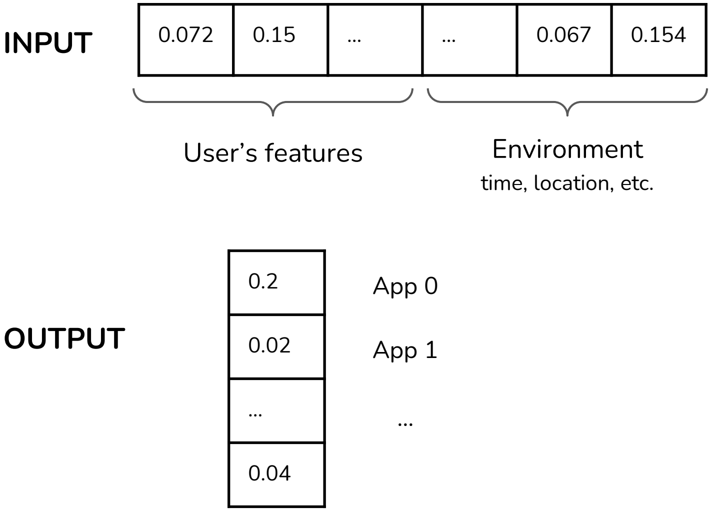
Model 2: Does this belong to class 2?

...

A problem can be framed as different task types

Problem: predict the app users will most likely open next

Classification



A problem can be framed as different task types

Problem: predict the app users will most likely open next

Classification

INPUT

0.072	0.15	...	...	0.067	0.154

$\underbrace{\qquad\qquad\qquad}_{\text{User's features}}$ $\underbrace{\qquad\qquad\qquad}_{\text{Environment}}$

User's features

Environment
time, location, etc.

OUTPUT

0.2	App 0
0.02	App 1
...	...
0.04	

⚠️ Every time an app is added/removed, you have to retrain your model ⚠️

Framing can make the problem easier/harder

Problem: predict the app users will most likely open next

Regression

										OUTPUT	
INPUT 0	0.072	0.15	...	...	0.067	0.154				0.03	**App 0**
INPUT 1	0.072	0.15	...	...	0.067	0.154				0.06	**App 1**
INPUT ...	0.072	0.15	...	...	0.067	0.154				0.25	**App ...**

User's features

Environment
time, location, etc.

App's features

Project objectives

- ML objectives
- Business objectives

Project objectives

- ML objectives
  - Performance
  - Latency
  - etc.

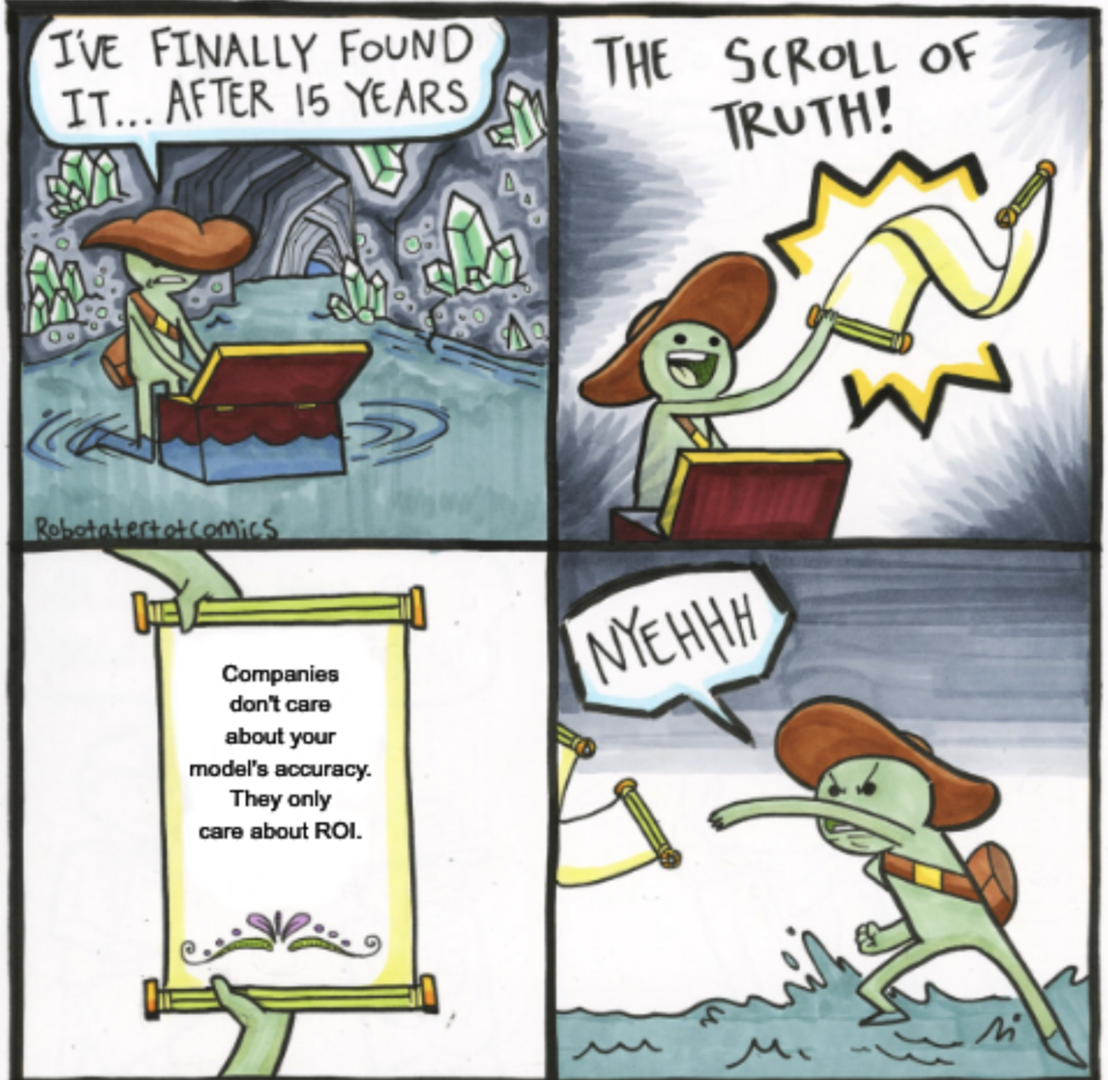
How to evaluate accuracy/F1/etc. without ground truth labels?

Project objectives

- ML objectives
  - Performance
  - Latency
  - etc.
- Business objectives
  - Cost
  - ROI
  - Regulation & compliance

Project objectives

- ML objectives
  - Performance
  - Latency
  - etc.
- Business objectives
  - Cost
  - ROI
  - Regulation & compliance



Business objectives

How can this ML project increase profits directly or indirectly?

- Directly: increasing sales (ads, conversion rates), cutting costs
- Indirectly: increasing customer satisfaction, increasing time spent on a website

ML <-> business: can be tricky

ML model gives customers more personalized solutions

customers happier

customers' problems solved faster

customers spending more money

customers spending less money

ML <-> business: mapping

- Baselines
  - Existing solutions, simple solutions, human experts, competitors solutions, etc.
- Usefulness threshold
  - Self-driving needs human-level performance. Predictive texting doesn't.
- False negatives vs. false positives
  - Covid screening: no false negative (patients with covid shouldn't be classified as no covid)
  - Fingerprint unlocking: no false positive (unauthorized people shouldn't be given access)
- Interpretability
  - Does it need to be interpretable? If yes, to whom?
- Confidence measurement (how confident it is about a prediction)
  - Does it need confidence measurement?
  - Is there a confidence threshold? What to do with predictions below that threshold—discard it, loop in humans, or ask for more information from users?

Constraints: time & budget

- Time
  - Rule of thumb: 20% time to get initial working system, 80% on iterative development
- Budget
  - Data, resources, talent

Time/budget tradeoffs

- Use more (powerful) machines
- Hire more people to label data faster
- Run more experiments in parallel
- Buy existing solutions

Constraints: privacy

- Annotation
  - Can data be shipped outside organizations for annotation?
- Storage
  - What kind of data are you allowed to store? How long can you store it?
- Third-party solutions
  - Can you share your data with a 3rd party (e.g. managed service)?
- Regulations
  - What regulations do you have to conform to?

Technical constraints

- Competitors
- Legacy systems

Four phases of ML adoption

Phase 1: Before ML

"If you think that machine learning will give you a 100% boost, then a heuristic will get you 50% of the way there."

Martin Zinkevich, Google

https://newsfeed.org/what-mark-zuckerbergs-news-feed-looked-like-in-2006/

Phase 2: Simplest ML models

Start with a simple model that allows visibility into its working to:

- validate hypothesis
- validate pipeline

Phase 3: Optimizing simple models

- Different objective functions
- Feature engineering
- More data
- Ensembling

Phase 4: Complex ML models

2. Decoupling objectives

Decoupling objectives

Possible high-level goals when building a ranking system for newsfeed?

1. minimize the spread of misinformation
2. maximize revenue from sponsored content
3. maximize engagement

Zoom poll: which goal would you choose?

Goal: maximize engagement

Step-by-step objectives:

1. Filter out spam
2. Filter out NSFW content
3. Rank posts by engagement: how likely users will click on them

Wholesome newsfeed

Goal: maximize users' engagement while minimizing the spread of extreme views and misinformation

Step-by-step objectives:

1. Filter out spam
2. Filter out NSFW content
3. Filter out misinformation
4. Rank posts by quality
5. Rank posts by engagement: how likely users will click on them

Decoupling objectives

Goal: maximize users' engagement while minimizing the spread of extreme views and misinformation

Step-by-step objectives:

1. Filter out spam
2. Filter out NSFW content
3. Filter out misinformation
4. Rank posts by quality
5. Rank posts by engagement: how likely users will click on it

How to rank posts by both quality & engagement?

Multiple objective optimization (MOO)

- Rank posts by quality
  - Predict posts' quality
  - Minimize **quality_loss:** difference between predicted quality and true quality

- Rank posts by how likely users will click on it
  - Predict posts' engagement
  - Minimize **engagement_loss**: difference between predicted clicks and true clicks

One model optimizing combined loss

- Rank posts by quality
  - Predict posts' quality
  - Minimize **quality_loss:** difference between predicted quality and true quality

- Rank posts by how likely users will click on it
  - Predict posts' engagement
  - Minimize **engagement_loss**: difference between predicted clicks and true clicks

$$\underbrace{\text{loss} = \alpha\ \text{quality\_loss} + \beta\ \text{engagement\_loss}}$$

Train one model to minimize this combined loss
Tune α and β to meet your need

Side note 1: check out Pareto optimization if you want to learn about how to choose α and β

One model optimizing combined loss

- Rank posts by quality
  - Predict posts' quality
  - Minimize **quality_loss:** difference between predicted quality and true quality

- Rank posts by how likely users will click on it
  - Predict posts' engagement
  - Minimize **engagement_loss**: difference between predicted clicks and true clicks

$$\underbrace{\text{loss} = \alpha \text{ quality\_loss} + \beta \text{ engagement\_loss}}_{\text{Train one model to minimize this combined loss}}$$

Side note 2: this is quite common, e.g. style transfer

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

A Neural Algorithm of Artistic Style (Gatys et al, 2017)

One model optimizing combined loss

- Rank posts by quality
  - Predict posts' quality
  - Minimize **quality_loss:** difference between predicted quality and true quality

- Rank posts by how likely users will click on it
  - Predict posts' engagement
  - Minimize **engagement_loss**: difference between predicted clicks and true clicks

$$\underbrace{\text{loss} = \alpha\ \text{quality\_loss} + \beta\ \text{engagement\_loss}}_{\text{Train one model to minimize this combined loss}}$$

⚠️ Every time you want to tweak α and β, you have to retrain your model! ⚠️

Multiple models: each optimizing one objective

- Rank posts by quality
  - Predict posts' quality
  - Minimize **quality_loss:** difference between predicted quality and true quality

- Rank posts by how likely users will click on it
  - Predict posts' engagement
  - Minimize **engagement_loss**: difference between predicted clicks and true clicks

$\mathbf{M_q}$: optimizes **quality_loss**
$\mathbf{M_e}$: optimizes **engagement_loss**

Rank posts by $\boldsymbol{\alpha}$ $\mathbf{M_q(post)} + \boldsymbol{\beta}\ \mathbf{M_e(post)}$

Now you can tweak α and β without retraining models

Decouple different objectives

- Easier for training:
  - Optimizing for one objective is easier than optimizing for multiple objectives
- Easier to tweak your system:
  - E.g. α % model optimized for quality + β % model optimized for engagement
- Easier for maintenance:
  - Different objectives might need different maintenance schedules
    - Spamming techniques evolve much faster than the way post quality is perceived
    - Spam filtering systems need updates more frequently than quality ranking systems

4. Data Engineering 101

Very basic. For details, take a database class!

Data engineering 101

- Data sources
- Data formats
- Data models
- Data storage engines & processing

Data sources

- User generated
- Systems generated
- Internal databases: users, inventory, customer relationships
- Third-party data

Data sources

Users generated data	Systems generated data
User inputs	Logs, metadata, predictions
Easily mal-formatted	Easier to standardize
Need to be processed ASAP	OK to process periodically (unless to detect problems ASAP)
	Can grow very large very quickly ● Many tools to process & analyze logs: Logstash, DataDog, Logz, etc. ● OK to delete when no longer useful

Users' behavioral data (clicks, time spent, etc.) is often system-generated but is considered **user data**

Third-party data: creepy but fascinating

- Types of data
  - social media, income, job
- Demographic group
  - men, age 25-34, work in tech
- More available with Mobile Advertiser ID
- Useful for learning features
  - people who like A also like B

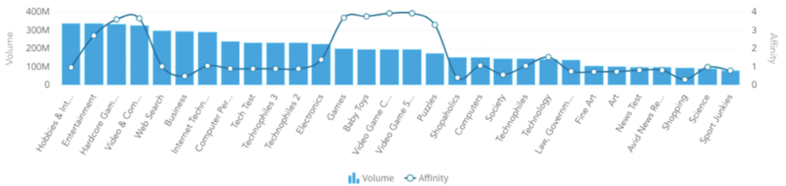
Top interests

—

They love computing and electronic entertainment. If you want to reach players, try targeting at their top interests.

Data point affinity and volume

Data point intersection volume and affinity, sorted by volume.



61 M profiles

Remote working

Millions of people decided to #stayhome and work remotely to limit the spread of coronavirus. Use our Remote working segment to easily reach them and show software or products that will help them stay effective.

How did we build the segment?

—

Our segment includes profiles of users who recently read articles, watched videos or used mobile apps which refers to:

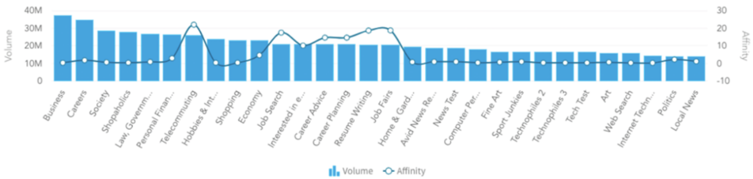
- remote working
- effective ways of working from home
- tools for remote workers
- homeschooling and e-learning

—

If you want to reach remote workers, try to extend your target group by selecting the top interests, which include Telecommuting, Career Planing or Personal Finance.

Data point affinity and volume

Data point intersection volume and affinity, sorted by volume.



onaudience.com/audience-data

How to store your data?

Storing your data is only interesting if you want to access it later

- Storing data: **ser**ialization
- Unloading data: **des**erialization

How to store your data?

Data formats are
agreed upon standards
to serialize your data so that
it can be transmitted & reconstructed later

Data formats: questions to consider

- How to store multimodal data?
  - `{‘image’: [[200,155,0], [255,255,255], ...], ‘label’: ‘car’, ‘id’: 1}`
- Access patterns
  - How frequently the data will be accessed?
- The hardware the data will be run on
  - Complex ML models on TPU/GPU/CPU

Data formats

Format	Binary/Text	Human-readable	Example use cases
JSON	Text	Yes	Everywhere
CSV	Text	Yes	Everywhere
Parquet	Binary	No	Hadoop, Amazon Redshift
Avro	Binary primary	No	Hadoop
Protobuf	Binary primary	No	Google, TensorFlow (TFRecord)
Pickle	Binary	No	Python, PyTorch serialization

Row-major

Column-major

Row-major vs. column-major

Column-major:
- stored and retrieved column-by-column
- good for accessing features

Row-major:
- stored and retrieved row-by-row
- good for accessing samples

	Column 1	Column 2	Column 3
Sample 1	…	…	…
Sample 2	…	…	…
Sample 3	…	…	…

Row-major vs. column-major: DataFrame vs. ndarray

Pandas DataFrame: column-major

- accessing a row much slower than accessing a column and NumPy

```
# Get the column `date`, 1000 loops
%timeit -n1000 df["Date"]

# Get the first row, 1000 loops
%timeit -n1000 df.iloc[0]
```

```
1.78 µs ± 167 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
145 µs ± 9.41 µs per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```

NumPy ndarray: row-major by default

- can specify to be column-based

```
df_np = df.to_numpy()
%timeit -n1000 df_np[0]
%timeit -n1000 df_np[:,0]
```

```
147 ns ± 1.54 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
204 ns ± 0.678 ns per loop (mean ± std. dev. of 7 runs, 1000 loops each)
```


https://github.com/chiphuyen/just-pandas-things

Text vs. binary formats

	Text files	**Binary files**
Examples	CSV, JSON	Parquet
Pros	Human readable	Compact
Store the number *1000000*?	7 characters -> 7 bytes	If stored as int32, only 4 bytes

You can unload the result of an Amazon Redshift query to your Amazon S3 data lake in Apache Parquet, an efficient open columnar storage format for analytics. Parquet format is up to 2x faster to unload and consumes up to 6x less storage in Amazon S3, compared with text formats. This enables you to save data transformation and enrichment you have done in

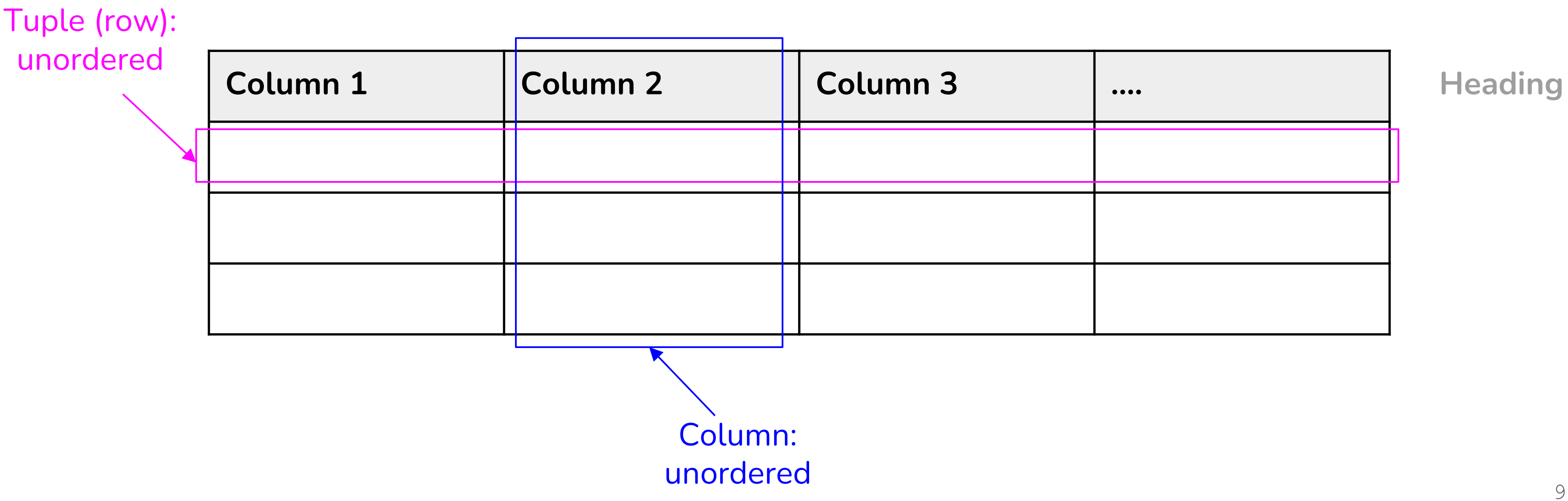
amazon REDSHIFT

Data models

- Describe how data is represented
- Two main paradigms:
  - Relational model
  - NoSQL

Relational model (est. 1970)

- Similar to SQL model
- Formats: CSV, Parquet



Relational model: normalization

What if we change “Banana Press” to “Pineapple Press”?

Title	Author	Format	Publisher	Country	Price
Harry Potter	J.K. Rowling	Paperback	Banana Press	UK	$20
Harry Potter	J.K. Rowling	E-book	Banana Press	UK	$10
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	$30
The Hobbit	J.R.R. Tolkien	Paperback	Banana Press	US	$30
Sherlock Holmes	Conan Doyle	Paperback	Guava Press	US	$15

Original Book Relation

Relational model: normalization

Title	Author	Format	Publisher ID	Price
Harry Potter	J.K. Rowling	Paperback	1	$20
Harry Potter	J.K. Rowling	E-book	1	$10
Sherlock Holmes	Conan Doyle	Paperback	2	$30
The Hobbit	J.R.R. Tolkien	Paperback	1	$30
Sherlock Holmes	Conan Doyle	Paperback	2	$15

Updated Book Relation

Publisher ID	Publisher	Country
1	Banana Press	UK
2	Guava Press	US

Publisher Relation

Relational model: normalization

Title	Author	Format	Publisher ID	Price
Harry Potter	J.K. Rowling	Paperback	1	$20
Harry Potter	J.K. Rowling	E-book	1	$10
Sherlock Holmes	Conan Doyle	Paperback	2	$30
The Hobbit	J.R.R. Tolkien	Paperback	1	$30
Sherlock Holmes	Conan Doyle	Paperback	2	$15

Publisher ID	Publisher	Country
1	Banana Press	UK
2	Guava Press	US

Pros:

- Less mistakes (standardized spelling)
- Easier to update
- Easier localization

Cons:

- Slow to join across multiple large tables

Relational Model & SQL Model

- SQL model slightly differs from relational model
  - e.g. SQL tables can contain row duplicates. True relations can't.
- SQL is a query language
  - How to specify the data that you want from a database
- SQL is declarative
  - You tell the data system what you want
  - It's up to the system to figure out how to execute
    - Query optimization

SQL

- SQL is an essential data scientists' tool

LEARN SQL!

NoSQL: No SQL -> Not Only SQL

- Document model
- Graph model

NoSQL

- Document model
  - Central concept: document
  - Relationships between documents are rare
- Graph model
  - Central concept: graph (nodes & edges)
  - Relationships are the priority

Document model: example

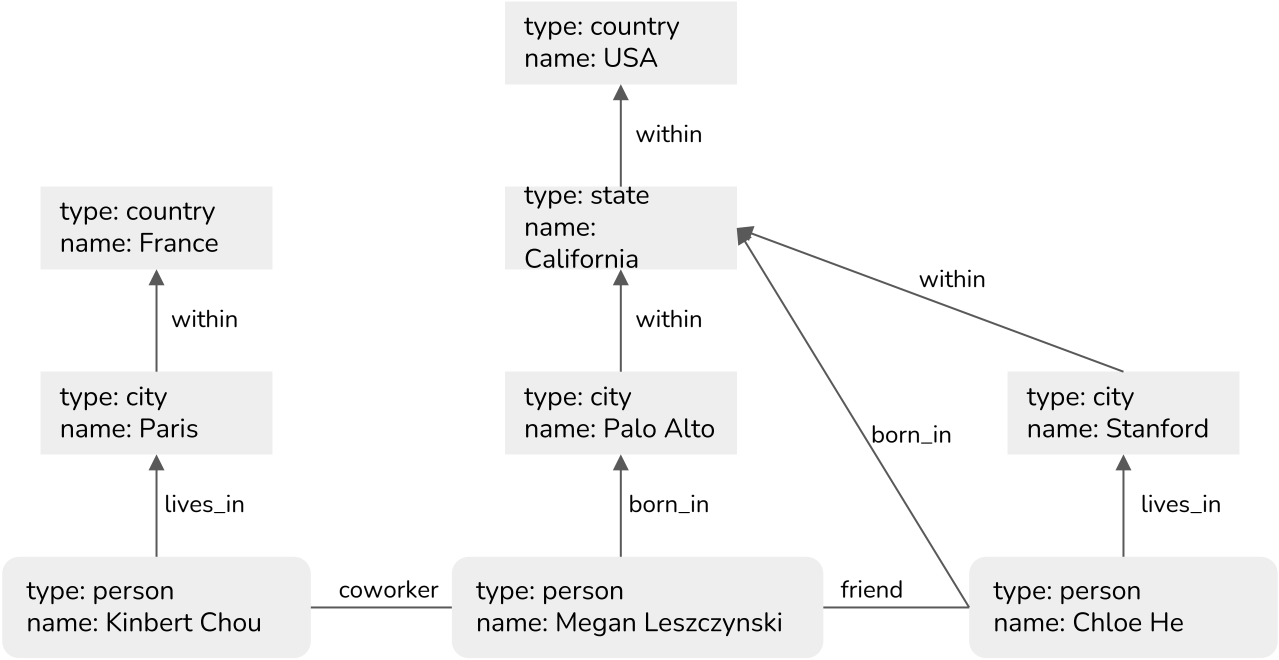
- Book data in the document model
- Each book is a document

```
# Document 1: harry_potter.json
{
    "Title": "Harry Potter",
    "Author": "J.K. Rowling",
    "Publisher": "Banana Press",
    "Country": "UK",
    "Sold as": [
        {"Format": "Paperback", "Price": "$20"},
        {"Format": "E-book", "Price": "$10"}
    ]
}

# Document 2: sherlock_holmes.json
{
    "Title": "Sherlock Holmes",
    "Author": "Conan Doyle",
    "Publisher": "Guava Press",
    "Country": "US",
    "Sold as": [
        {"Format": "Paperback", "Price": "$30"},
        {"Format": "E-book", "Price": "$15"}
    ]
}

# Document 3: the_hobbit.json
{
    "Title": "The Hobbit",
    "Author": "J.R.R. Tolkien",
    "Publisher": "Banana Press",
    "Country": "UK",
    "Sold as": [
        {"Format": "Paperback", "Price": "$30"},
    ]
}
```

Graph model

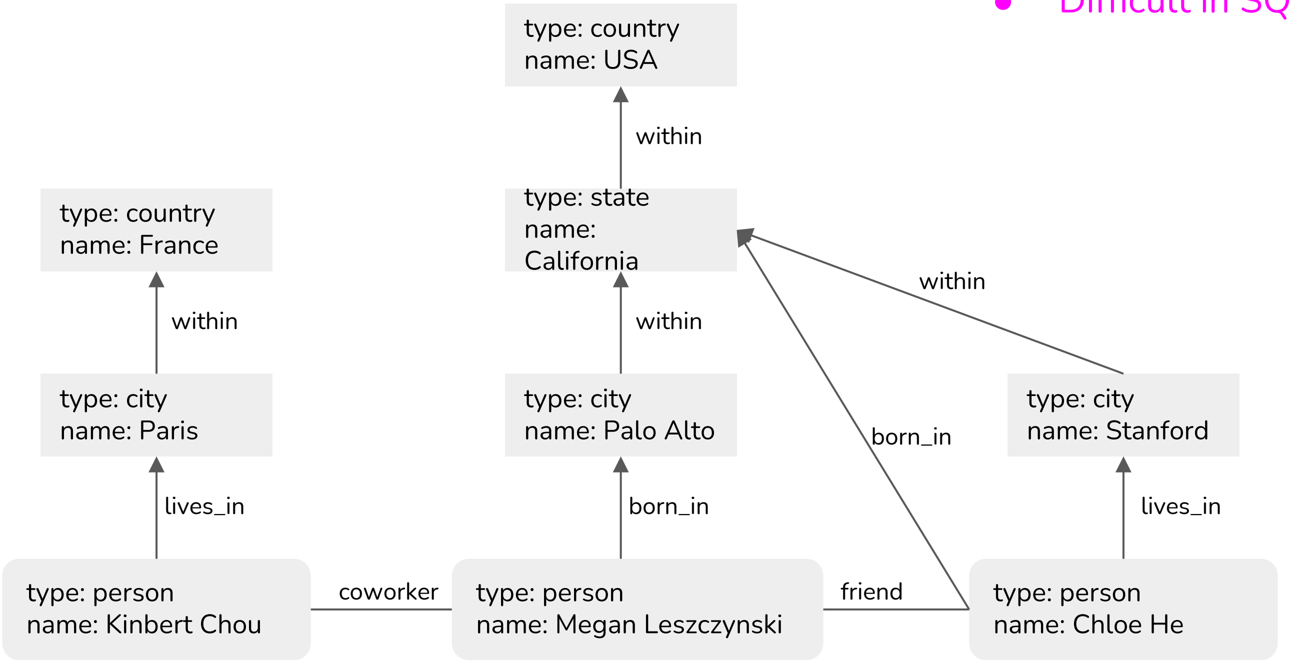


type: country
name: USA
within
type: country
name: France
type: state
name:
California
within
within
within
type: city
name: Paris
type: city
name: Palo Alto
born_in
type: city
name: Stanford
lives_in
born_in
lives_in
type: person
name: Kinbert Chou
coworker
type: person
name: Megan Leszczynski
friend
type: person
name: Chloe He

Graph model

Query: show me everyone who was born in the USA?

- Easy in graph
- Difficult in SQL



Structured vs. unstructured data

Structured	Unstructured
Schema clearly defined	Whatever
Easy to search and analyze	Fast arrival (e.g. no need to clean up first)
Can only handle data with specific schema	Can handle data from any source
Schema changes will cause a lot of trouble	No need to worry about schema changes
Data warehouses	Data lakes

Structured vs. unstructured data

Structured	Unstructured
Structure is assumed at **write**	Structure is assumed at **read**

Data Storage Engines & Processing

Databases optimized for

Transactional processing

Analytical processing

OnLine Transaction Processing (OLTP)

- Transactions: tweeting, ordering a Lyft, uploading a new model, etc.
- Operations:
  - Insert when generated
  - Occasional update/delete

OnLine Transaction Processing

- Transactions: tweeting, ordering a Lyft, uploading a new model, etc.
- Operations:
  - Inserted when generated
  - Occasional update/delete
- Requirements
  - Low latency
  - High availability

OnLine Transaction Processing

- Transactions: tweeting, ordering a Lyft, uploading a new model, etc.
- Operations:
  - Inserted when generated
  - Occasional update/delete
- Requirements
  - Low latency
  - High availability
  - ACID not necessary
    - **Atomicity**: all the steps in a transaction fail or succeed as a group
      - If payment fails, don't assign a driver
    - **Isolation**: concurrent transactions happen as if sequential
      - Don't assign the same driver to two different requests that happen at the same time

See ACID:
Atomicity,
Consistency,
Isolation,
Durability

OnLine Transaction Processing

- Transactions: tweeting, ordering a Lyft, uploading a new model, etc.
- Operations:
  - Inserted when generated
  - Occasional update/delete
- Requirements
  - Low latency
  - High availability
- Typically row-major

Row

```
INSERT INTO RideTable(RideID, Username, DriverID, City, Month, Price)
VALUES ('10', 'memelord', '3932839', 'Stanford', 'July', '20.4');
```

OnLine Analytical Processing (OLAP)

- How to get aggregated information from a large amount of data?
  - e.g. what's the average ride price last month for riders at Stanford?
- Operations:
  - Mostly SELECT

OnLine Analytical Processing

- Analytical queries: aggregated information from a large amount of data?
    - e.g. what's the average ride price last month for riders at Stanford?
- Operations:
    - Mostly SELECT
- Requirements:
    - Can handle complex queries on large volumes of data
    - Okay response time (seconds, minutes, even hours)

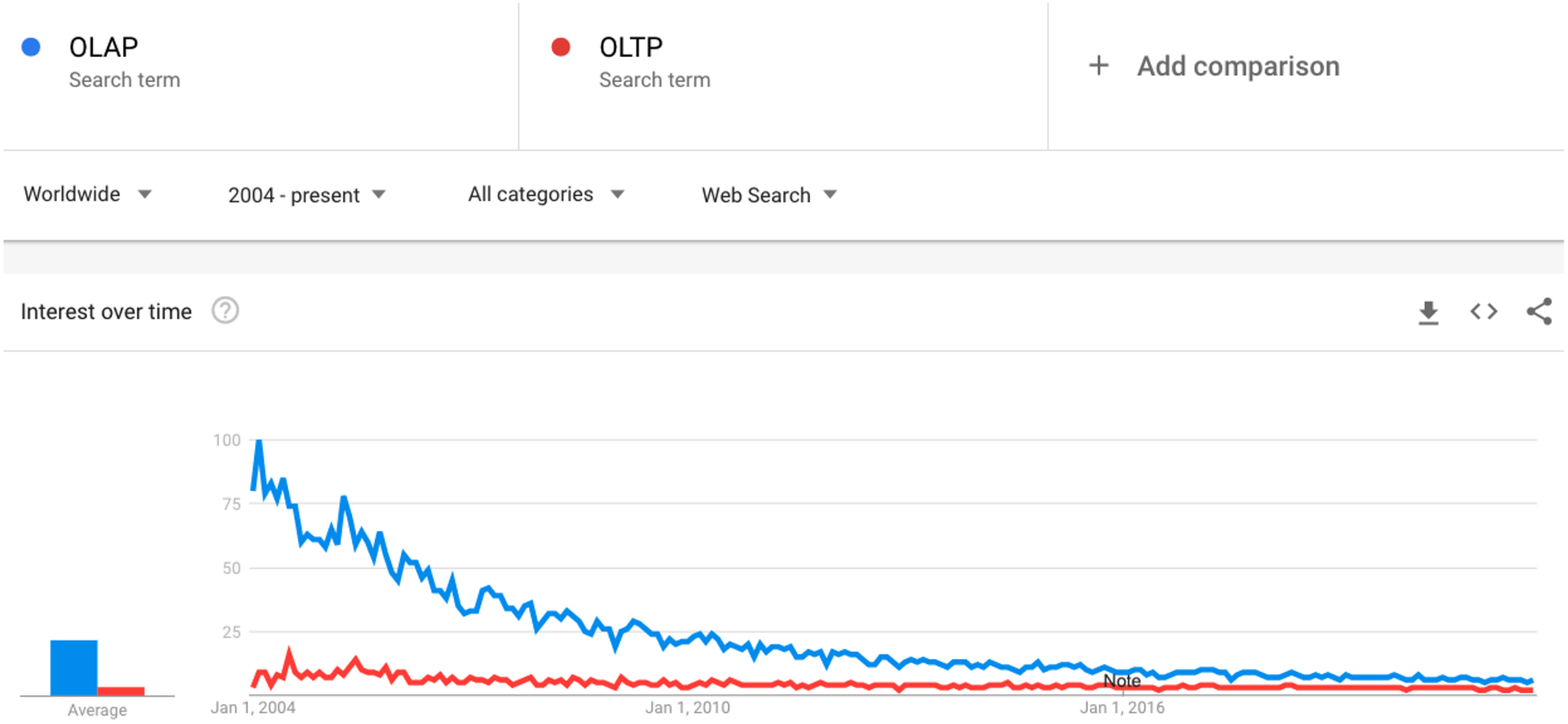
OnLine Analytical Processing

- Analytical queries: aggregated information from a large amount of data?
  - e.g. what's the average ride price last month for riders at Stanford?
- Operations:
  - Mostly SELECT
- Requirements:
  - Can handle complex queries on large volumes of data
  - Okay response time (seconds, minutes, even hours)
- Typically column-major

Column

```
SELECT AVG(Price)
FROM RideTable
WHERE City = 'Stanford' AND Month = 'July';
```

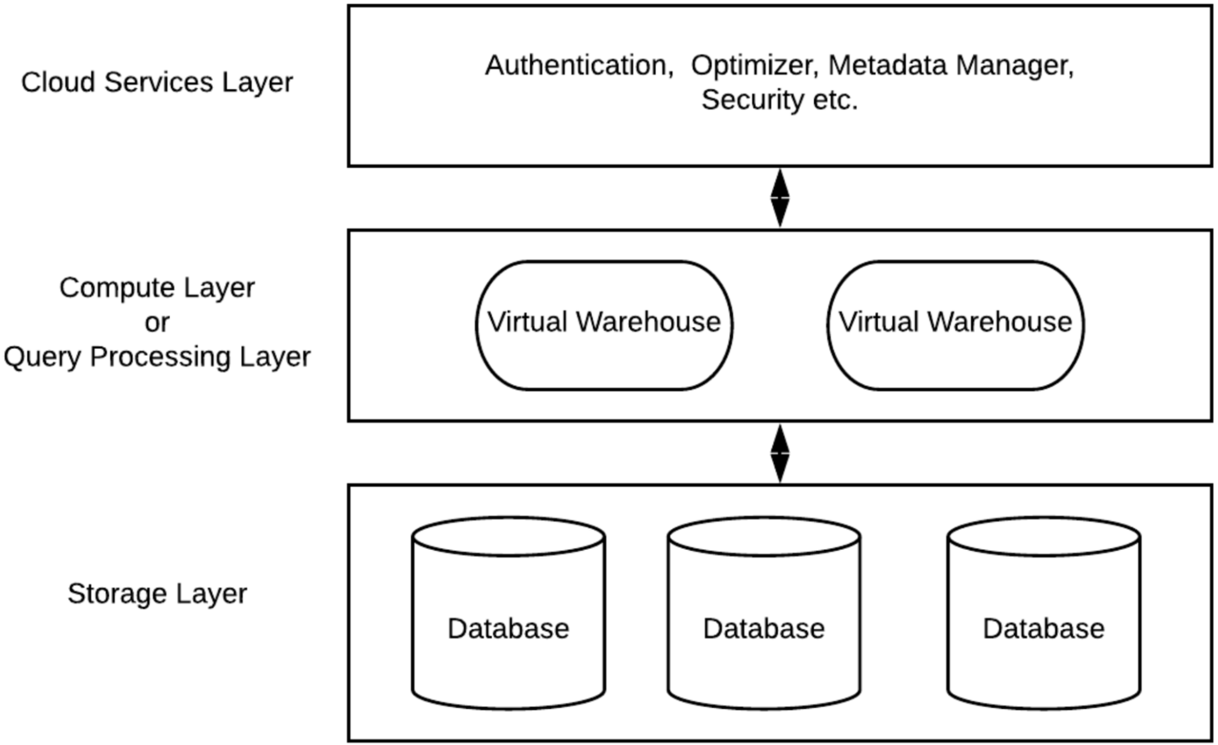
OLTP & OLAP are outdated terms



Decoupling storage & processing

- OLTP & OLAP: how data is stored is also how it's processed
  - Same data being stored in multiple databases
  - Each uses a different processing engine for different query types
- New paradigm: storage is decoupled from processing
  - Data can be stored in the same place
  - A processing layer on top that can be optimized for different query types

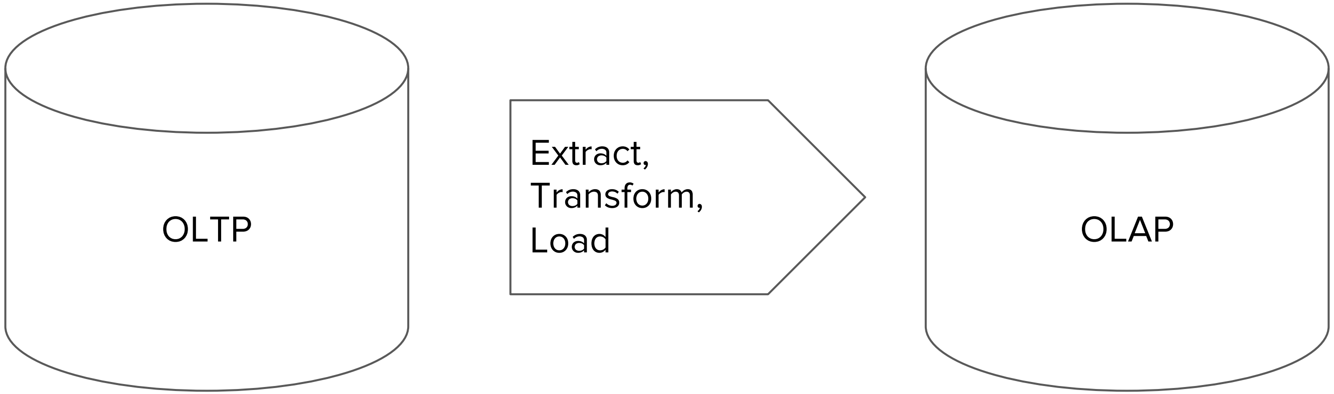
Decoupling storage & processing



https://hevodata.com/blog/snowflake-architecture-cloud-data-warehouse/

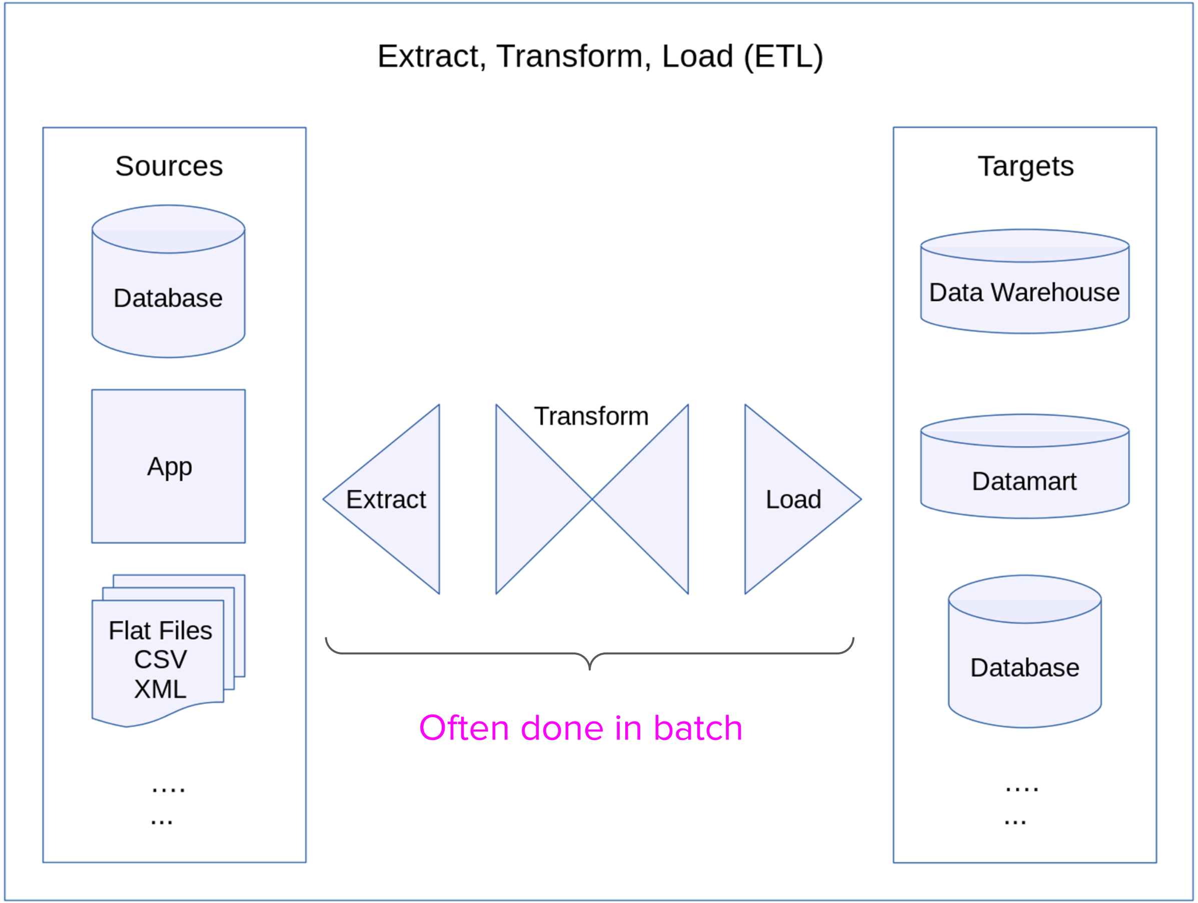

ETL
ETL EVERYWHERE
LIGHTYEAR

ETL (Extract, Transform, Load)



Transform: the meaty part

- cleaning, validating, transposing, deriving values, joining from multiple sources, deduplicating, splitting, aggregating, etc.



https://commons.wikimedia.org/wiki/File:Extract,_Transform,_Load_Data_Flow_Diagram.svg

ETL -> ELT

Structured -> unstructured -> structured

want more flexibility tools & infra

standardized

ETL -> ELT -> ETL